CPS2: Embedded Systems
Credits: 3
Course number: (Special Topics 6501/department equivalent)
Instructor: John Lach

Description
This is a core Cyber Physical Systems (CPS) class. Embedded systems are special-purpose computers that are often tightly coupled with electronic and physical components and must operate within real-time performance, battery, and size constraints. By the end of the course, the students will be able to:

- Understand the fundamentals of embedded system architectures and real-time operating systems.
- Develop embedded C/C++ programs in real-time operating systems.
- Explain how memory management, interrupt handling, thread management, task scheduling, and software/hardware interfacing are done in real-time operating systems.
- Design an embedded system based on a given spec and validate if the functional requirements are satisfied.

Course Outline:
Introduction
- Background on Cyber Physical Systems
- Overview of Design Process
  - Modeling dynamic behaviors (Joint dynamics)
  - Design of embedded computational platforms => This course
  - Analysis of system based on specs/properties (Logic, model checking, quantitative analysis)

Part 1 – Embedded System Architectures
- Sensors and Actuators
  - Sensors: Sound, pressure, force, temperature, distance, flow, acceleration, light,
  - Actuators: Mechanical, electrical, thermal, sound, light
- Embedded Processors
  - Microcontrollers, FPGA, ASIC, GPU, DSP
- Memory Architectures
  - Technologies (RAM, ROM, EEPROM, Flash)
  - Memory Hierarchy (Memory Maps, Reg File, Caches, Stack)
  - Memory allocation and management (Static and dynamic)
  - Addressing Modes
- Embedded Input and Output (I/O)
  - Software Hardware Interfaces (Parallel, Serial, Time, Analog)
  - Interrupts (Interrupt controllers, context switch, periodic timers)
  - I/O Programming (Memory mapped I/O, race conditions)
  - Serial I/O (UART, SSI)
  - Analog I/O (DAC and ADC)
  - Sampling, Digitization, Nyquist theorem, Timers, Timers, Pulse Width Modulation

Part 2 – Embedded Software Programming
- Assembly Language
  - Addressing modes, operands, memory access, function calls
- C Programming
Review of C
Tool chains (Compiler, Assembler, Debugger, Programmer)
Pointers and memory allocation

- **Real-time Operating Systems**
  - Multi-tasking (Thread management)
  - Scheduling (Real-time constraints, Example scheduling algorithms)
  - Synchronization/communication among tasks
  - Critical sections and semaphores (Deadlocks, Starvation, Priority Inversion)
  - Examples of real-world RTOS (for IoT and sensor networks)

**References**